import pandas as pd

try:

    df = pd.read\_csv('twitter\_training.csv')

    display(df.head())

except FileNotFoundError:

    print("Error: 'twitter\_training.csv' not found.")

except pd.errors.ParserError:

    print("Error: Could not parse the CSV file. Check the file format.")

except Exception as e:

    print(f"An unexpected error occurred: {e}")

# Display data types of each column

print(df.dtypes)

# Check for missing values

print(df.isnull().sum())

# Analyze the distribution of the target variable

print(df['Positive'].value\_counts())

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))

df['Positive'].value\_counts().plot(kind='bar')

plt.title('Distribution of Sentiment')

plt.xlabel('Sentiment')

plt.ylabel('Count')

plt.show()

# Address missing values in the tweet text column.

# Removing rows with missing tweet text as they are not useful for sentiment analysis.

original\_rows = len(df)

df.dropna(subset=['im getting on borderlands and i will murder you all ,'], inplace=True)

dropped\_rows = original\_rows - len(df)

print(f"Dropped {dropped\_rows} rows with missing tweet text.")

# Check for and remove duplicate rows.

duplicate\_rows = df.duplicated().sum()

df.drop\_duplicates(inplace=True)

print(f"Removed {duplicate\_rows} duplicate rows.")

# Display first 5 rows of the cleaned DataFrame.

display(df.head())

import nltk

nltk.download('punkt\_tab')

import re

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

def clean\_text(text):

    text = str(text).lower()

    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation

    words = nltk.word\_tokenize(text)

    words = [w for w in words if not w in stop\_words]  # Remove stop words

    words = [stemmer.stem(w) for w in words]  # Stemming

    return ' '.join(words)

df['cleaned\_text'] = df['im getting on borderlands and i will murder you all ,'].apply(clean\_text)

display(df.head())

from sklearn.feature\_extraction.text import TfidfVectorizer

# Initialize TfidfVectorizer

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000, max\_df=0.95, min\_df=2)

# Fit and transform the cleaned text

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['cleaned\_text'])

# Convert the sparse matrix to a dense array

tfidf\_matrix = tfidf\_matrix.toarray()

from sklearn.model\_selection import train\_test\_split

# Assuming 'Positive' is the target variable and 'tfidf\_matrix' is already defined.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    tfidf\_matrix, df['Positive'], test\_size=0.2, random\_state=42, stratify=df['Positive']

)

from sklearn.linear\_model import LogisticRegression

# Initialize and train the Logistic Regression model

model = LogisticRegression(max\_iter=1000, solver='saga')

model.fit(X\_train, y\_train)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Predict on the test data

y\_pred = model.predict(X\_test)

# Calculate evaluation metrics

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted', zero\_division=0)

recall = recall\_score(y\_test, y\_pred, average='weighted', zero\_division=0)

f1 = f1\_score(y\_test, y\_pred, average='weighted', zero\_division=0)

print(f"Accuracy: {accuracy:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1-score: {f1:.4f}")

# Compute the confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

# Visualize the confusion matrix using a heatmap

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

            xticklabels=model.classes\_, yticklabels=model.classes\_)

plt.title('Confusion Matrix')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()